# StablePlaza: An efficient constant product AMM for stable tokens based on anchored liquidity

v1.0 - Jazzer9F

2022/07/11

**Abstract**

This paper introduces StablePlaza, which is an efficient distributed exchange for stable tokens. The main innovation used in StablePlaza is the concept of anchored liquidity which is a specific case of concentrated liquidity using an amount of virtual liquidity tailored specifically for stable coins. The maximum price difference between tokens is set based on historic data for stable token relative prices. Anchored liquidity is generalized to multi-token AMMs and the first deployment of StablePlaza will list the 4 largest USD stable tokens on Ethereum. The result is a stable token exchange which is highly efficient regarding from both the capital deployment and gas consumption perspectives.

## 1 Introduction

### 1.1 Rise of the automated market maker

Since the pioneering work of Bancor [2, 3], Gnosis [1] and Uniswap [4, 5, 6], automated market makers (AMMs) have grown to become one of the main pillars of Decentralized Finance (DeFi) on distributed networks. The most popular AMMs allow trading along an invariant curve and are known as Constant Function Market Makers (CFMM). In comparison to the high complexity strategies that Professional Market Makers (PMM) use on centralized exchanges the CFMM trading strategy is remarkably simple. Despite their complete inability to distinguish between informed order flow and uninformed order flow, they typically (for serious projects/tokens) still manage to be profitable for liquidity providers in the long run.

CFMMs are uniquely suitable for implementation in smart contracts since their trading strategy is public and can be implemented efficiently. Recent years have seen explosive growth of the capital locked in CFMMs on public blockchains.

### 1.2 Constant product AMMs

The most common form of constant product market makers is the constant product market maker which was popularized by UniSwap [4]. In it's basic form it simply dictates that the product of the number of tokens in the exchange is kept constant across trades. For a pool with two tokens as used in Uniswap we get:

$$x \cdot y = k \tag{1}$$

The swap invariant can be generalized to more tokens as pioneered by Balancer who are operating pools with multiple tokens. The DefiPlaza project, of which StablePlaza is part, developed and deployed a highly optimized multi-token constant product AMM with a 16 of the most popular tokens in DeFi pooled together [7, 8]. Besides the ease of implementation in smart contracts, the main advantage of constant product AMMs is that they can absorb very large price swings and still function adequately. With large price swings not being uncommon in crypto currencies, this property is highly desirable for AMMs.

There are two main downsides to constant product AMMs. The first one is that the liquidity providers are exposed to impermanent loss risk. The total value of holdings in the pool in dollar equivalent terms always falls with respect to the dollar equivalent value of the same portfolio that is simply held. This loss may become quite significant in case of high relative price fluctuations of pooled tokens.

The second downside is that for typical daily price swings in the order of 2% the vast majority of the capital sits idly in the pool. For a typical trading day the trading volume for a typical pool is in the order of 10% of the capital locked in that pool. That means it takes 10 days for any given token to be traded, which

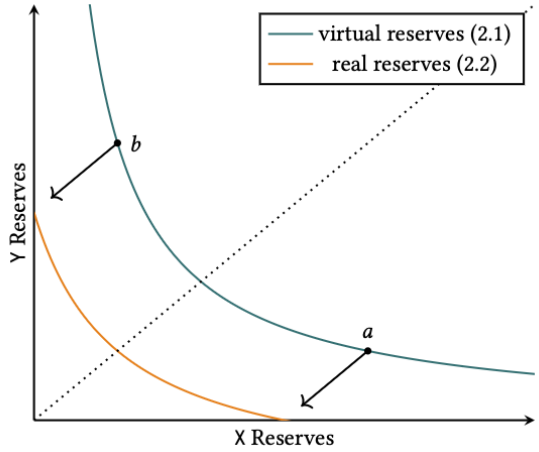


Figure 1: Introduces the concepts of real/virtual reserves and concentrated liquidity, copied from figure 2 of the Uniswap V3 whitepaper.

is well below typical trading volumes of 100% of deployed capital that PMMs would see on centralized exchanges.

## 1.3 Stable coin specific CFMMs

When there is more information about the expected price of tokens in a pool, that information can be used to increase the trading efficiency of capital deployed in a pool. If we know that the price of all tokens is exactly equal, we could use a constant sum AMM and trade tokens on a 1:1 basis. However, if there is even the slightest price imbalance a constant sum pool would run out of liquidity for some of its tokens. Even for hard pegged tokens slight imbalances are observed in reality making a constant sum AMM impractical. The Curve team pioneered the StableSwap algorithm [9] which combines the constant sum invariant with the constant product invariant with special weights as below:

$$An^n \sum x_i + D = ADn^n + \frac{D^{n+1}}{n^n \prod x_i} \tag{2}$$

The StableSwap algorithm uses a tuning parameter $A$ called the "amplification coefficient" which is used to tune the overal shape of the swap invariant. The algorithm works by first calculating the value $D$ (the amount of each token when all prices are equal) and then ensuring the invariant is maintained across trades. The main advantage for the algorithm is that while it concentrates the capital around the price equality it does allows to keep offering liquidity if the price for any of the tokens goes outside of the expected range. The main disadvantage is that it requires iterative calculation which is inconvenient in an environment where computation is as expensive as Ethereum.

## 1.4 Concentrated liquidity

As an alternative approach to increase capital efficiency Uniswap pioneered the concept of 'concentrated liquidity' in its v3 release of May 2021 [6]. The idea of concentrated liquidity is that liquidity providers can select a price range $[a, b]$ on the constant product hyperbole and their liquidity would only be active when the price is within that range. If the price moves outside of the range, the position is reduced to containing only the cheapest asset. Figure 1 is taken from the Uniswap paper.

Concentrated liquidity allows greatly improved capital efficiency, ie liquidity providers to earn significant more trading fees as long as they correctly estimate the range for the relative prices that the tokens will be trading at. To implement this Uniswap elected to use price ticks spaced every $0.01\%$ of relative price change, discretizing the options for $a$ and $b$. For each liquidity position $[a, b]$ Uniswap V3 follows the invariant below:

$$(x + \frac{L}{\sqrt{p_b}})(y + L\sqrt{p_a}) = L^2 \tag{3}$$

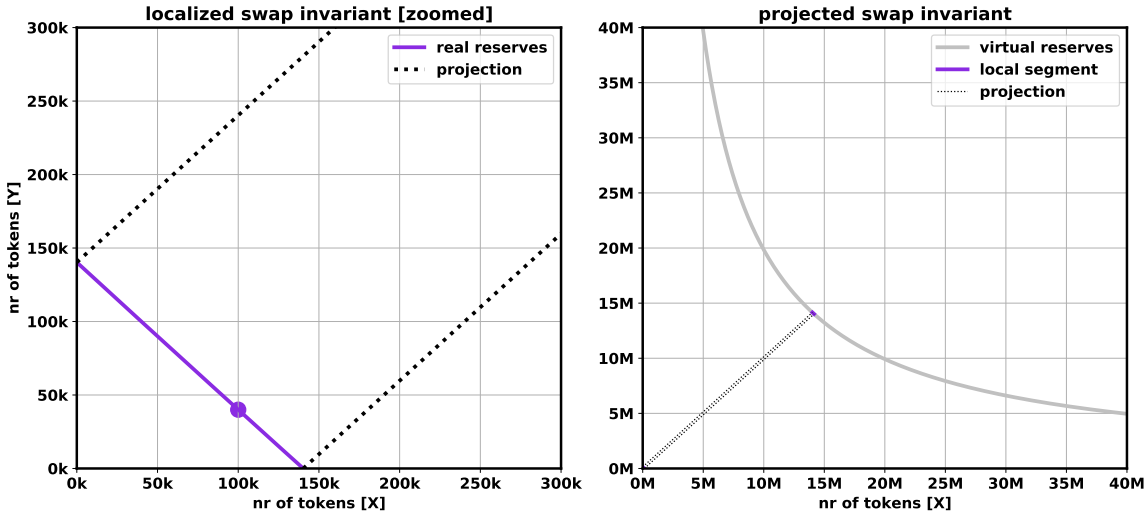


Figure 2: Anchored liquidity principle. Left) real reserves exchange curve. Right) virtual reserves exchange curve. The virtual reserves curve is chosen so as to anchor the localized exchange rate around 1.

In this equation, $L$ is the amount of liquidity for this position, and $p_a$, $p_b$ are the relative prices bounding the range. The invariant used is effectively the same as the constant product invariant with virtual liquidity added to the each of the tokens. The concentrated liquidity concept applies to all token pairs but work especially well for stable tokens where the prices should return to parity unless one of the tokens breaks its peg. The main disadvantage of concentrated liquidity as implemented by Uniswap is that the concept of a unified LP token is lost as liquidity positions need to be tracked individually. Moreover it's more expensive gas wise than Uniswap V2 to execute swaps, especially if tick boundaries are crossed.

# 2 The anchored liquidity concept

The method to increasing capital efficiency proposed in this paper is to follow the concentrated liquidity approach of Uniswap V3 [6], with hard-coded $[a, b]$ price ranges for a given pool and extended to support multi-token pools. By hard-coding the $[a, b]$ range all liquidity positions become equal such that LP tokens can once again be used to represent shares of the pool liquidity. We propose the term 'anchored liquidity' to describe this strategy. See the figure below for a 2-token example where there are 100k X tokens in the pool and 40k Y tokens. Figure 2 shows this principle.

Anchored liquidity is a special case of the virtual reserves as introduced by Uniswap. The $a$ and $b$ are chosen in such a way that we have $\sqrt{p_a} = 1/\sqrt{p_b}$. This approach symmetrically anchors the relative price range around parity and results in both $x$ and $y$ getting offset by the same amount. By selecting appropriate values for $a, b$ the localized swap invariant becomes an almost, but not quite, straight line. The name 'anchored liquidity' is chosen because the amount of virtual liquidity is set specifically in such a way that the price ratio between the tokens is anchored to a certain range regardless of how many reserves are held by the exchange for each token.

## 2.1 Multi-token anchored liquidity

The anchored liquidity principle can be extended to multi-token AMMs. Just like with the two-token version, we set the same offset on each token. If we name this offset $\Delta$, we get the following constant product invariant:

$$\prod_{i=1..N} (x_i + \Delta) = k \tag{4}$$

We should select $\Delta$ in such a way that the price range covered is appropriate for stable tokens. Since the summer of 2021, stable token prices have been within relative prices of $\pm 0.5\%$ the vast majority of the time. Note that in equation 4, if any token has real reserves of zero it has virtual reserves of $\Delta$. Say a maximum price differential of $2\%$ is targeted between normal operation (in balance) and zero real reserves

for any token. Assuming the exchange is perfectly balanced and all $x_i$ are equal to $\bar{x}$ the following equality holds:

$$(\bar{x} + \Delta)^N = 1.02\Delta^N \tag{5}$$

Note that $\Delta$ doesn't change when a swap is executed along the constant product invariant. Thus, we can substitute the right hand side of the special case where the exchange is in balance from equation 5 into equation 4 yielding equation 6.

$$\prod_{i=1..N} (x_i + \Delta) = 1.02\Delta^N \tag{6}$$

Equation 6 has $N$ solutions for $\Delta$ in general. It can be proven that only one of these solutions is a positive real number, and that is the solution we'll be using throughout the rest of the paper. We'll be using the notation $\hat{\Delta}$ to refer to this solution. Using this notation, the anchored liquidity swap invariant is now written in the form of equation 7.

$$\prod_{i=1..N} \left(x_i + \hat{\Delta}\right) = k \tag{7}$$

## 2.2 Token swaps

A swap is an exchange of tokens between an external party and StablePlaza that respects the invariant as given in equation 7. If a swap is executed exactly along the swap invariant, the value for $k$ doesn't change. Thus, using superscript $0$ to indicate original situation and superscript $1$ to indicate post-swap variables we can write:

$$\prod_{i=1..N} \left(x_i^0 + \hat{\Delta}\right) = \prod_{i=1..N} \left(x_i^1 + \hat{\Delta}\right) \tag{8}$$

While the invariant is calculated across all $N$ listed tokens, in practice a swap is always executed between two of these tokens. Without loss of generality we may look at any two tokens. Since the other $N - 1$ token reserves aren't impacted they appear on both sides of the equation and cancel each other out, yielding equation 9.

$$\left(x_1^0 + \hat{\Delta}\right)\left(x_2^0 + \hat{\Delta}\right) = \left(x_1^1 + \hat{\Delta}\right)\left(x_2^1 + \hat{\Delta}\right) \tag{9}$$

Once again without loss of generality we can assume that $x_1$ is the token the external party sends into the exchange, where $x_2$ is received in return. Then, writing $dx_i$ to indicate change in reserves for a particular token we arrive at equation 10.

$$\left(x_1^0 + \hat{\Delta}\right)\left(x_2^0 + \hat{\Delta}\right) = \left(x_1^0 + dx_1 + \hat{\Delta}\right)\left(x_2^0 - dx_2 + \hat{\Delta}\right) \tag{10}$$

Equation 10 may be reduced to equation 11 through simplification and rearrangement of variables to describe the number of output tokens received by the external party as a function of the number of input tokens.

$$dx_2 = \frac{dx_1}{\left(x_1^1 + \hat{\Delta}\right)}\left(x_2^0 + \hat{\Delta}\right) \tag{11}$$

The resulting amount of output tokens is the simple ratio of input tokens to new virtual reserves of $x_1$ multiplied by virtual reserves of the output token. Note that equation 11 can efficiently be implemented in smart contracts.

## 2.3 Adding and removing liquidity

Liquidity in the exchange is represented by LP tokens. When liquidity is added or removed we should mint or burn the correct amount of LP tokens. Using $L$ as variable to indicate liquidity as in equation 3 we can generalize to the anchored liquidity version in equation 12.

$$\prod_{i=1..N} \left(x_i + \hat{\Delta}\right) = L^N \tag{12}$$

Rearranging terms yields the liquidity invariant as in equation 13.

$$\frac{L^N}{\prod_{i=1..N}\left(x_i + \hat{\Delta}\right)} = 1 \tag{13}$$

Note that the virtual liquidity $\hat{\Delta}$ changes when the liquidity in the exchange changes. If we add or remove liquidity exactly along the invariant we arrive at equation 14 describing liquidity change operations.

$$\frac{{L^0}^N}{\prod_{i=1..N}\left(x_i^0 + \hat{\Delta}^0\right)} = \frac{{L^1}^N}{\prod_{i=1..N}\left(x_i^1 + \hat{\Delta}^1\right)} \tag{14}$$

Combining equations 14 and 6 the liquidity change equation can be simplified to equation 15.

$$L^0\hat{\Delta}^1 = L^1\hat{\Delta}^0 \tag{15}$$

This final liquidity change equation clearly shows the fundamental relation between LP tokens and virtual reserves in a short and elegant formula. However, to apply this formula in a smart contract environment we need to calculate $\hat{\Delta}$ which is not straight forward to do. Therefore, when moving to an actual implementation of anchored liquidity we may need to consider simplifications.

# 3 The StablePlaza implementation

This section discusses how the concept of anchored liquidity is used to build an efficient stable token exchange. The objective is to have the high capital efficiency of the anchored liquidity concept, implemented in a way that makes it highly competitive from a gas cost perspective. This approach stays true to the DefiPlaza philosophy of offering users the lowest cost trades available on Ethereum L1.

To implement anchored liquidity in a smart contract AMM, we need to solve equation 6 for $\Delta$. Solving the equation entails finding the roots of an $N$-th order polynomial, which is considered easy to do in modern computers but is quite expensive to do on-chain in Ethereum. For the main StablePlaza exchange, $N$ will be equal to $4$. For an exchange like StablePlaza that is explicitly targeting low-gas consumption we need to simplify things further.

## 3.1 Approximating $\hat{\Delta}$ efficiently

If we consider the special case where the reserves for all tokens held by the exchange are equal we get equation 5, which can be solved for $\Delta$ yielding:

$$\hat{\Delta} = \frac{1}{1.02^{\frac{1}{N}} - 1}\bar{x} \tag{16}$$

$$\approx 201.49\bar{x} \tag{17}$$

Thus, if the exchange is completely balanced the offset value $\hat{\Delta}$ that gives rise to the desired token price range is a bit more than 200 times the real reserves of each single token. It is helpful to rewrite the solution of equation 16 as follows:

$$\hat{\Delta} = \frac{1}{N\left(1.02^{\frac{1}{N}} - 1\right)} \sum_{i=1..N} \bar{x} \tag{18}$$

$$\approx \frac{201.49}{N} \sum_{i=1..N} \bar{x} \tag{19}$$

The formulation of equation 18 views $\hat{\Delta}$ as a multiplier times the sum of all reserves held by the exchange. Equation 18 gives the multiplier for perfect balance in the exchange, which is the upper bound for the multiplier between total real reserves and $\hat{\Delta}$. The lower bound for the multiplier occurs when the exchange is maximally imbalanced, i.e. all reserves are zero except for one. In that case all $x_i$ are equal to zero except for one which is denoted by $x^*$ and 6 reduces to the below.

$$(x^* + \Delta)\Delta^{N-1} = 1.02\Delta^N \tag{20}$$

Solving for $\Delta$ and rewriting as a multiplier times the sum of the exchange gives us the following equation 21.

$$\hat{\Delta} = 50 \sum_{i=1..N} x_i \tag{21}$$

For StablePlaza, we can fill in $N = 4$ and find an upper bound on the multiplier of $50.37$. Thus, we know that for any given amount of exchange liquidity, the value of $\hat{\Delta}$ lies between $50$ and $50.37$ times the total liquidity in the exchange. So although it is expensive to compute the exact solution for $\hat{\Delta}$ on-chain, we can efficiently compute a decent approximate solution by summing the total liquidity on the exchange and multiplying by a constant.

When an approximation is used for $\hat{\Delta}$ the impact is that the actual maximum price range for swaps is slightly different from the target price range, but the difference is small enough that it shouldn't matter in practice. The larger concern is that with the same amount of real reserves in the exchange we now can observe for slightly different amounts of virtual reserves depending on what the balance between the tokens is when the value for $\hat{\Delta}$ is calculated. If one is not careful with this approximation, users could find ways to abuse it to extract slightly more tokens from the exchange than they should be able to. It is critical that the approximation is safe in the sense that it cannot be abused to drain liquidity from the exchange. To ensure the approximation is always on the conservative side, StablePlaza uses the lowest approximation of the multiplier (50) since using a slightly lower value for $\Delta$ will result in slightly higher than perfect price impact for users such that the exchange liquidity is always protected.

The approximation of $\hat{\Delta}$ by a constant times the sum of real liquidity in the exchange is denoted by $\tilde{\Delta}$ to allow the reader to distinguish between the exact and the approximate solutions.

## 3.2 Adding liquidity

StablePlaza is initialised with an equal amount of LP tokens as USD tokens initially held as real reserves. Over time, users can add and remove liquidity to the pool. Ideally the anchored liquidity invariant would be implemented as per equation 15. However, solving the exact invariant is difficult since we can't efficiently compute the exact solution for $\hat{\Delta}^0$ and $\hat{\Delta}^1$ on chain.

Unfortunately, if we use a linear approximation for $\hat{\Delta}$ as discussed above equation 15 becomes linear in $L$ which means that the amount of LP tokens minted or burnt becomes directly related to the tokens added / removed without any consideration for relative scarcity. All tokens would be treated equally when it comes to liquidity add/remove even though they are not treated equally when it comes to swaps which opens up an attack vector. Thus, the linear approximation for $\Delta$ can't be used for liquidity add/remove operations. Thus, a different method is required.

Liquidity add/remove operations on StablePlaza are single sided, effectively representing a trade between one of the listed tokens and the LP token. Mathematically a single sided liquidity add is modelled as if the user added liquidity to all the pool tokens in the required ratios and then swapped at zero fee into the token that is actually added. The total number of LP tokens that gets minted is determined by the ratio of new liquidity to current total liquidity. Suppose that an amount of $x_{in}$ is added to the exchange for a particular token:

$$R_{in} = \frac{x_{in}}{x^0 + \tilde{\Delta}^0} \tag{22}$$

$$dLP = \left((1 + R_{in})^{\frac{1}{N}} - 1\right) \cdot LP_0 \tag{23}$$

The ratio $R_{in}$ is the ratio of added liquidity to existing virtual liquidity for that token. The original fraction of LP token liquidity is calculated with an N-ways inverse swap along the constant product curves. Note that the total amount of liquidity tokens $LP_0$ used in 23 includes LP tokens representing the real liquidity as well as virtual LP tokens representing the virtual liquidity across all tokens. It turns out that the total LP ($LP_0$) can be written as a multiplier times the real LP ($LP_x$). If there is only one token with real liquidity (maximum imbalance) then the total liquidity can be calculated with equation 25.

$$LP_0 = LP_x + LP_\Delta \tag{24}$$

$$= LP_x + 200LP_x = 201LP_x \tag{25}$$

Equations 23 and 25 can be implemented in smart contracts efficiently. StablePlaza uses equation 23 to calculate the LPs minted when liquidity is added to the exchange with the factor $201$ to represent the virtual liquidity. Note that the virtual swap operations are done maintaining the value for $\tilde{\Delta}$ before the liquidity was added. This guarantees fairness to existing LP holders. More accurate would be to mint LP and updating $\Delta$ along the curve as liquidity is added, but this is not practicable in smart contract implementation.

## 3.3 Removing liquidity

Similar to the liquidity add, removing liquidity from the exchange is also a single-sided operation. A fraction of all LP tokens are burnt which releasing that same fraction of each of the (virtual) reserves. These tokens are then swapped at zero fee into the token that is requested as output token. The final amount of output tokens is thus given by the ratio of tokens (again including virtual LP tokens) released multiplied by the factor accounting for the swaps back into the requested token. Equation 27 is used to describe the amount of tokens released for a given amount of LP tokens:

$$R_{out} = \frac{LP_{burnt}}{LP_0^*} \tag{26}$$

$$x_{out} = \left(1 - (1 - R_{out})^N\right)\left(x + \tilde{\Delta}\right) \tag{27}$$

Again, the virtual swaps are executed with $\tilde{\Delta}$ as it was before the liquidity remove. This way we can guarantee fairness towards other liquidity providers. Note that the amount of virtual LP tokens $LP_0^*$ is slightly different from the one used for adding liquidity. The reason is that while $201$ is the correct multiplier for a maximally imbalanced exchange, the multiplier is higher for a more balanced exchange. For liquidity add, using a lower multiplier is conservative. When removing liquidity, a higher multiplier is conservative.

The highest the multiplier can go is when the exchange is completely in balance (all token reserves are equal). In that case, the virtual liquidity can be calculated and from that the total liquidity $LP_0^*$ which can be proven to follow equation 30.

$$LP_0^* = LP_x + LP_\Delta \tag{28}$$

$$= LP_x + \frac{1}{1.02^{\frac{1}{4}} - 1} LP_x \tag{29}$$

$$\approx 202.49 LP_x \tag{30}$$

The exact level of (im)balance in the exchange cannot easily be calculated on chain. Thus, to guarantee conservatism with respect to liquidity add/remove transactions, LP tokens are always minted as if the exchange is completely skewed towards one token, while liquidity is removed as if the exchange were perfectly balanced. This allows an efficient implementation while guaranteeing that individual add/remove operations are fair towards the other liquidity providers. The cost of this approximation is that adding liquidity towards a given token and immediately removing it again incurs an $0.74\%$ penalty. This penalty can be reduced or even completely avoided (converted into a bonus) by adding liquidity when a token is under-represented and removing when the token is over-represented.

## 3.4 StablePlaza implementation details

The following subsections will discuss some of the choices made for StablePlaza to optimize the implementation.

### 3.4.1 Packing reserves array

To efficiently allow flash loans, flash swaps and potential UniswapV2 pair interface compatibility, StablePlaza needs to store the token reserves. The Ethereum Virtual Machine (EVM) that ultimately runs the StablePlaza bytecode is a 256 bit machine. This means that all basic operations are implemented using 256 bit variables. The most popular USD stable tokens have 6 decimals, so they are accurate to one ten-thousandths of a cent. This accuracy is more than enough for real world applications. With just 6 decimals, a 64 bit storage slot still allows values up to 18 trillion dollar to be stored, which should be enough for

StablePlaza. By reducing accuracy of all token reserves to 6 decimals even if a token natively has higher accuracy we can store all reserves in 64 bits each, which allows all four reserves to fit neatly in one 256 storage slot. Therefore, all four token reserves are updated with just a single write to the blockchain saving gas.

#### 3.4.2 Efficiently implementing liquidity add/remove

Solving equations 23 and 27 requires raising a variable to the $1/N$-th power and $N$-th power respectively. By selecting and hardcoding $N = 4$ these computions can be implemented more efficiently. Raising to the $1/N$-th power in particular depends on a numerical approximation, for which it turns out that if we use a power of two for $N$ many coefficients combine into powers of two as well and we can use save some gas by combining multiple operations into a single bitshift in several places.

#### 3.4.3 When to update $\Delta$

When there is no transaction fee, performing a swap does not impact the virtual reserves curve and the $\Delta$ value need not be updated after a swap. In practice there will be a transaction fee implemented which slightly increases the real reserves held by the exchange over time. Technically, whenever the reserves change away from the existing virtual liquidity curve the value for $\Delta$ should be updated as well. However, updating $\Delta$ would constitute a write to the blockchain which is an expensive operation. Not updating $\Delta$ after a swap fee is levied means that the value of $\Delta$ will be slightly smaller than it would be if we had updated it. Using slightly too small value for $\Delta$ leads to slightly higher price impact from trades but is not a safety concern. Since the expected impact of temporarily not updating $\Delta$ is very small, we consider the benefit of saving a write to the blockchain to outweigh the issue of slightly higher price impact. Thus, StablePlaza only updates the value for $\Delta$ when liquidity is added or removed from the exchange.

### 3.5 Earning a share of fees through staking

StablePlaza has been built to include a staking feature to seamlessly integrate it into the DefiPlaza ecosystem. Where other protocols use an admin fee, in StablePlaza part of the fee is allocated to users who stake the DefiPlaza governance token DFP2. This ensures that this part of the fees flows back to the community and incentivizes community members to provide liquidity and volume for the StablePlaza exchange.

### 3.6 Listed tokens

De-pegging of stable coins is the risk which is front and center in DeFi since the recent implosion of UST. StablePlaza is no different from other stable token exchanges in that it is designed to work for tokens which are trading roughly at parity. Impermanent loss is neglible as long as the prices remain close to parity, but increases significantly if one of the tokens were to loose control of the peg. The first StablePlaza will list four major stable tokens. To minimize risk of tokens de-pegging, algorithmic stable tokens are avoided completely. The following tokens will be listed in the exchange: USDC, USDT, DAI and BUSD. These tokens are currently among the largest, most reputable and long-lasting stable tokens striking what we believe to be the best balance between risk and reward.

Once StablePlaza is running, it will still be possible to change out the listed tokens though timelocked admin transactions. The number of tokens listed on the exchange is fixed to 4.

## 4 Results

The StablePlaza pool was loaded onto an Ethereum mainnet fork for testing and gas consumption comparison for the main operations. To ensure this is an apples to apples comparison, the starting situation (amount of tokens, token approval) are set identically. Table 1 below compares the gas results on a Ganache copy of mainnet running hardfork Istanbul.

The swap represents a swap from DAI to USDT, where the user spends all his DAI and doesn't hold have any USDT yet. The liquidity add represents a single sided DAI liquidity add, where the user spends all their DAI and doesn't hold LP tokens yet. The liquidity remove is a remove to DAI, where the user withdraws all LP tokens into DAI and doesn't hold any DAI yet. StablePlaza is about 20k gas cheaper on

| | StablePlaza | Curve 3pool |
|---|---|---|
| DAI → USDT | 94,180 | 113,570 |
| DAI → LP | 88,249 | 159,188 |
| LP → DAI | 81,515 | 121,559 |

Table 1: Gas cost for identical operations compared on mainnet fork

a single token exchange transaction, and significantly more than that for single-sided liquidity add/remove transactions.

Although it will be possible to change listed tokens, the total number of listed tokens is hard coded to 4. In the future, we expect that the liquidity in StablePlaza may be used to build metapools similar to the ones currently implemented by Curve. We don't have an exact gas estimate for swaps against underlying for those metapools yet, but given that the liquidity add/remove transactions are significantly cheaper than their Curve equivalents, we expect to offer highly competitive gas costs for metapool operations. This whitepaper will be updated once the metapool contract is ready and numbers become available.

# References

[1] Alan Lu. Building a decentralized exchange in ethereum. https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e, 2017.

[2] Eyal Hertzog, Guy Benartzi, Galia Benartzi. Continuous liquidity and asynchronous price discovery for tokens through their smart contracts;aka smart tokens. https://whitepaper.io/document/52/bancor-whitepaper, 2017.

[3] Eyal Hertzog, Guy Benartzi, Galia Benartzi, Omri Ross. Continuous liquidity for cryptographic tokens through their smart contracts. https://storage.googleapis.com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en.pdf, 2018.

[4] Hayden Adams. Uniswap whitepaper. https://hackmd.io/@HaydenAdams/HJ9jLsfTz, 2018.

[5] Hayden Adams, Noah Zinsmeister, Dan Robinson. Uniswap v2 core. https://uniswap.org/whitepaper.pdf, 2020.

[6] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, Dan Robinson. Uniswap v3 core. https://uniswap.org/whitepaper-v3.pdf, 2021.

[7] Jazzer9F. Defi plaza — a low cost dex for ethereum. https://jazzerradix.medium.com/defi-plaza-a-low-cost-dex-for-ethereum-45b10c1ea2a0, 2021.

[8] Jazzer9F. Defiplaza: Making defi on ethereum affordable again. https://jazzerradix.medium.com/defiplaza-making-defi-on-ethereum-affordable-again-b533724a0885, 2021.

[9] Michael Egorov. Stableswap - efficient mechanism for stablecoin liquidity. https://curve.fi/files/stableswap-paper.pdf, 2019.